

Weniger ist mehr

Was die anstehende Version TLS 1.3 bringt

Das kurz vor der Verabschiedung stehende Verschlüsselungsprotokoll TLS 1.3 könnte sich als großer Wurf erweisen. Denn anders als bei den Vorgängerversionen, deren Einführung sich über viele Jahre hinzog, bringt eine möglichst schnelle Umstellung konkrete Vorteile in Bezug auf Sicherheit und Performance.

Von Jürgen Schmidt

Transport Layer Security, kurz TLS, ist der Standard für das zusätzliche Sicherheit signalisierende „S“ in HTTPS, IMAPS und vielen weiteren Protokollen. Wegen TLS kann man sicher sein, dass etwa eine Online-Überweisung wirklich nur bei der eigenen Bank ankommt und unterwegs aus 100 Euro an die Tante nicht plötzlich 1000 Euro in die Ukraine werden.

Die letzten Versionen brachten eigentlich nur Detailverbesserungen, änderten aber wenig an der grundsätzlichen Funktionsweise. Dafür sammelten sich in den rund zwanzig Jahren seit der Einführung damals noch unter dem Namen SSL unzählige Zusatzfunktionen und Optionen an. Sie machten das ganze Protokoll zu einem unüberschaubaren und letztlich unsicheren Misthaufen, der wegen der heiligen Kuh der Rückwärtskompatibilität immer weiter anwuchs.

Das große Ausmisten

Damit sollte jetzt Schluss sein. Deshalb stellte die Arbeitsgruppe für die neue TLS-Version jede einzelne Funktion und auch viele Design-Entscheidungen erneut auf den Prüfstand und klopfte sie auf Nutzen und Gefahren für die Sicherheit ab.

Als Resultat gingen einige prominente Elemente von TLS über den Jordan. Das bekannteste Opfer ist RSA: Bei einer Ver-

bindung gemäß TLS 1.3 darf der Austausch des (symmetrischen) Sitzungsschlüssels nicht mehr über RSA erfolgen. Dem liegt die Erkenntnis zu Grunde, dass Diffie-Hellman-Verfahren nicht nennenswert mehr Aufwand erfordern, aber zusätzlich sicherstellen können, dass der Verlust eines Serverschlüssels keine Auswirkungen auf bereits in der Vergangenheit verschlüsselte Inhalte hat (Forward Secrecy). Zukünftig muss der Schlüsselaustausch also zwingend via Diffie-Hellman erfolgen – bevorzugt über dessen Variante auf elliptischen Kurven namens ECDHE (siehe Kasten auf S. 174).

Das Ende der Erbsünde

Neben dem Schlüsselaustausch sind Integritätssicherung und natürlich die eigentliche Verschlüsselung zentrale Aufgaben von TLS. Die Urväter des Protokolls entschieden sich damals mangels konkreter Erkenntnisse über Vor- und Nachteile

eher zufällig für genau diese Reihenfolge: Erst wird ein Prüfwert über den Klartext gebildet (der Message Authentication Code, MAC) und an die Daten angehängt; erst dann wird verschlüsselt.

Dieses MAC-then-Encrypt erwies sich jedoch später als fataler Design-Fehler. Denn es erlaubte immer wieder Angriffe auf die Verschlüsselung, die etwas vereinfacht darauf beruhten, dass man versucht, Byte für Byte den Klartext zu erraten und die Integritätssicherung dem Angreifer anschließend verrät, ob er mit seiner Vermutung richtig lag (das sogenannte Padding-Orakel).

Das musste nach erfolgreicher Demonstration von Angriffen mehrfach provisorisch gefixt werden. In TLS 1.2 hat man zwar einen Zustand erreicht, von dem man glaubt, dass er wohl nicht mehr angreifbar sein sollte – so ganz sicher ist sich jedoch niemand.

Bis heute schleppt TLS diese Erbsünde des MAC-then-Encrypt-Problems mit sich herum. TLS 1.3 räumt damit endlich auf und fordert zwingend die sogenannte „Authenticated Encryption“, die Integritätssicherung mit Verschlüsselung in einem Schritt kombiniert.

Das bekannteste Authenticated-Encryption-Verfahren, der Galois Counter Mode (GCM), ist zwar bereits für TLS 1.2 definiert. Er führte jedoch lange Zeit ein Dasein im Schatten des Cipher Block Chaining (CBC), das neben dem MAC-then-Encrypt noch weitere Probleme aufweist. Damit ist jetzt endgültig Schluss: TLS 1.3 erlaubt kein CBC mehr.

Ebenfalls dem Großreinemachen geopfert wurden eine Reihe selten genutzter, aber problematischer Optionen wie Kompression und Neuaushandlung (Renegotiation) einer existierenden Verbindung. Die bekannt kaputten Verfahren wie Export Ciphers, MD5, SHA1, RC4? Ebenfalls weg! Dafür beherrscht TLS 1.3 die längst überfällige DJB-Kurve Curve-22519 und Goldilocks für Kryptografie auf Basis elliptischer Kurven (ECC).

Downgrades verhindern

Über das Abspecken hinaus hat die Arbeitsgruppe auch einige vorbeugende Härtnungsmaßnahmen ergriffen, um sich gegen kommende Angriffe zu wappnen. So waren bislang große Teile des Verbindungsaufbaus nicht kryptografisch gesi-

chert. Deshalb konnte etwa ein Man-in-the-Middle die vom Client an den Server gesendete Liste der unterstützten Krypto-Verfahren manipulieren, ohne dass Server oder Client diesen Eingriff bemerkten.

Ein Angreifer könnte somit diese Liste auf das eine Verfahren kürzen, das er brechen kann. Er erzwingt damit die Nutzung eines unsicheren, nur als „immer noch besser als gar nichts“ aufgeführten Verfahrens, obwohl eigentlich beide Kommunikationspartner sichere Verfahren beherrschen. Das ist keineswegs nur theoretisch; die TLS-Angriffe FREAK und Logjam nutzten diese Schwäche ganz real aus.

Bei TLS 1.3 wird der Verbindungsaufbau soweit irgend möglich digital signiert, was unter anderem Manipulationen an der Liste der unterstützten Verschlüsselungsverfahren unmöglich macht. Außerdem werden Server, die TLS 1.3 bereits beherrschen, einem Client einen Protokoll-Downgrade mit der hexadezimalen Zeichenfolge „44 4F 57 4E 47 52 44“ signalisieren, die nach ASCII übersetzt die Buchstaben DOWNGRD ergibt. Ein ebenfalls TLS-1.3-fähiger Client kann daran erkennen, dass er keineswegs einen veralteten Server vor sich hat, sondern dass gerade etwas schief läuft.

Weniger Overhead

Nach dem längst fälligen Ausmisten und allgemeinem Härten sind Performance-Verbesserungen die dritte wichtige Bau-

stelle von TLS 1.3. Im Zeitalter von Hardware, die das Ver- und Entschlüsseln mit AES nativ und damit sehr performant unterstützt, ist der Aufbau einer gesicherten Verbindung die wichtigste Bremse.

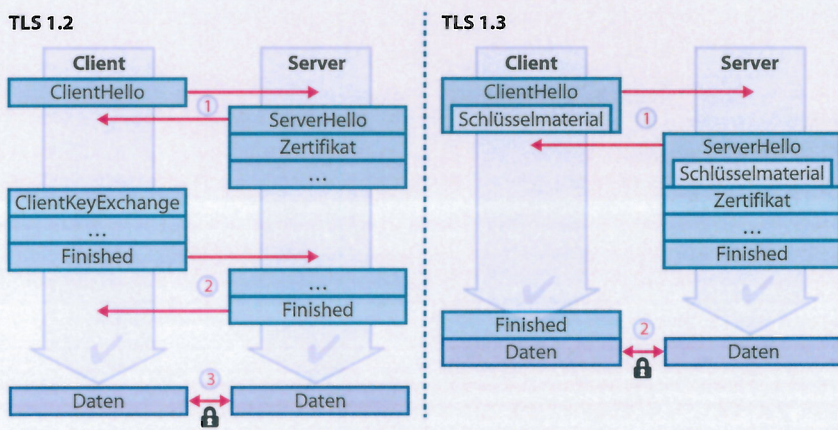
Auch hier hat sich im Prinzip seit den SSL-Ursprüngen wenig geändert: Eine normale TLS-Verbindung erfordert den Austausch von Verwaltungsinformationen und deren Bestätigung, sodass letztlich zwei komplette Kommunikationsschritte (Roundtrips) erfolgen, bevor die eigentliche Arbeit losgehen kann. Erst im dritten Roundtrip können Client und Server tatsächlich Nutzdaten austauschen. Je nach Verbindungsqualität bedeutet das bis zu eine Sekunde Wartezeit.

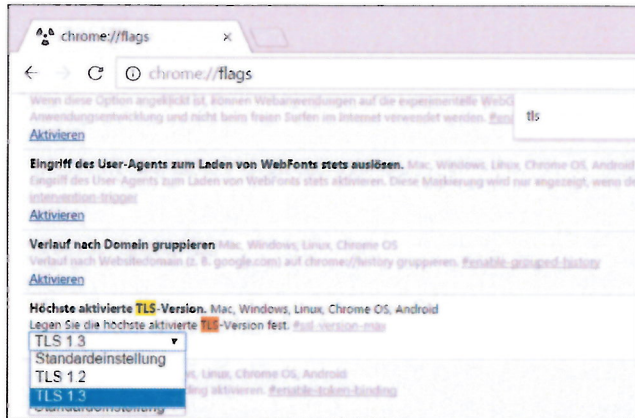
TLS 1.3 reduziert das im Regelfall auf einen Vorab-Roundtrip; schon im zweiten Schritt gehen wieder Nutzdaten über die Leitung. Im Wesentlichen funktioniert das so, dass der Client vorab schon plausible Annahmen über Verfahren macht, die der angesprochene Server hoffentlich unterstützt, und auch bereits dazu passendes Schlüsselmaterial auf Verdacht mitsendet. Das spart einen kompletten Roundtrip und damit die Hälfte des Verwaltungs-Overheads ein.

Dieser beschleunigte Aufbau funktioniert bereits bei der ersten Verbindung zu einem Server. Analog zur bereits in TLS 1.2 definierten Wiederaufnahme einer Verbindung (Session Resumption) mit nur einem Verwaltungsschritt enthält auch TLS 1.3

Schneller Verbindungsaufbau

TLS 1.3 reduziert den Overhead. Im Idealfall können die Kommunikationspartner bereits im zweiten Schritt gesicherte Daten austauschen.






Der aktuelle Chrome 56 beherrscht TLS 1.3 bereits; allerdings muss man es derzeit noch von Hand aktivieren.

(<https://tswg.github.io/tls13-spec/>). Eine Vorlage bei der verantwortlichen Internet Engineering Steering Group war für Ende Januar (nach Redaktionsschluss) geplant. Sollte nichts Unvorhergesehenes geschehen, kann eine Verabschiedung durchaus schon im Frühjahr 2017 erfolgen.

Es gibt auch bereits eine Reihe von Software-Projekten, die TLS 1.3 unterstützen. So beherrschen es Chrome 56 und die Entwickler-Version von Firefox bereits. In die Bibliotheken NSS und BoringSSL hat es ebenfalls schon Einzug gehalten; OpenSSL arbeitet an einer Umsetzung. Und CloudFlare bietet seinen Kunden bereits an, TLS 1.3 im Testbetrieb für die eigenen Seiten zu aktivieren.

Angesichts der wichtigen Verbesserungen ist zu hoffen, dass dem Standard nicht das Schicksal seines Vorgängers droht. TLS 1.2 wurde bereits 2008 verabschiedet, fand aber bis 2014 keine nennenswerte Verbreitung, was vor allem darauf zurückzuführen war, dass niemand so recht wusste, wozu man es wirklich braucht. Bei TLS 1.3 hingegen ist die Mission klar und von anerkannt hoher Dringlichkeit: Wir wollen ein sichereres Internet. (ju@ct.de) 

einen neuen Wiederaufnahmemechanismus, der optional sogar schon im allerersten Paket Nutzdaten transportieren kann.

Dabei benutzt der Client einen bei der letzten Verbindung mit dem Server vereinbarten Schlüssel (Preshared Key). Das bedeutet, dass die zusätzliche Verschlüsselung der Kommunikation quasi ohne zeitliche Verzögerung zustande käme. Allerdings bringt dieser sogenannte O-RTT-Modus die Gefahr von Replay-Attacken mit sich und ist somit Beschränkungen unterworfen. So darf er etwa nur für Kommunikation verwendet werden, die den Status auf dem Server nicht ändert. Also darf das erste Paket etwa eine HTTPS-Anfrage ent-

halten, die die Homepage des Servers abrufen, nicht aber einen Bezahlvorgang auslösen. Offenbar war die Performance der Arbeitsgruppe ausreichend wichtig, von ihrer generellen Richtung hin zu mehr Sicherheit etwas abzuweichen. Die Details dazu sind einer der wenigen Punkte, an denen aktuell noch gefeilt wird. Ob und wie das dann real zum Einsatz kommt, muss sich in der Praxis zeigen.

Der Stand der Dinge

Der Entwurf eines RFC für TLS 1.3 befindet sich derzeit in der Phase des letzten Feinschliffs. Die aktuelle Version wird bei Github vorgehalten und gepflegt

Banken protestieren und Server streiken – zwei Anekdoten am Rande

Das Mehr an Sicherheit durch das Weglassen unsicherer Optionen in TLS 1.3 führte nicht nur zu Zustimmung – im Gegenteil: Aus eher unerwarteter Richtung kam sogar Protest. So meldete sich im Herbst 2016 ein Interessenvertreter der Finanzdienstleister zu Wort, dass beim aktuellen Stand der Dinge die Banken ihren Verpflichtungen zur Kontrolle und Sicherung des internen Datenverkehrs nicht mehr nachkommen könne. Konkret: Damit sie auch in Zukunft alles mitlesen können, was in ihren Netzen passiert, solle man doch das Verbot des Schlüsselaustausches mit RSA in TLS 1.3 noch mal überdenken. Erfrischend war die spontane Antwort von Kenny Paterson: „Meine Einschätzung zur Anfrage: Nein. Begründung: Wir versuchen gerade, das Internet sicherer zu machen.“

Letztlich mündete die Diskussion dann in einen Internet Draft zu „Data Center use of Static Diffie-Hellman in TLS 1.3“.

Darin erläutert Matthew Green, wie man in einer definierten Umgebung Überwachung mit Hilfe fester Diffie-Hellman-Schlüssel umsetzen kann, ganz ohne dem Rest des Internet einen faulen Sicherheitskompromiss mit RSA aufzuzwingen.

Versions-Schmiere

Tests mit Vorabversionen von TLS 1.3 förderten ein weiteres unerwartetes Problem zu Tage. Ein signifikanter Teil aller Webserver lehnte Verbindungen kategorisch ab, in denen der Client angab, er könne bereits TLS 1.3. Laut Spezifikation muss der Server in einer solchen Situation eine niedrigere, von ihm bereits unterstützte TLS-Version wählen. Statt dessen brechen etwa 3 Prozent der Server den Verbindungsaufbau ab – zu viel, um es einfach zu ignorieren und die damit beim Anwender auftretenden Fehler in Kauf zu nehmen. Also musste für die Versionsaushandlung von TLS 1.3 ein

hässlicher Workaround umgesetzt werden, der auch diese kaputten Server bei Laune hält.

TLS 1.3 friert deshalb die interne Versionsnummer bei 1.2 (konkret: 0x0303) ein; darüber hinaus gibt es eine Liste mit allen unterstützten Versionen; der Empfänger wählt daraus die von ihm bevorzugte und muss alle ihm unbekannten Versionen ignorieren. Als Lehre für die Zukunft diskutieren die TLS-Architekten außerdem eine Maßnahme, die dafür sorgt, dass unbekannte Versionen tatsächlich regelmäßig auftauchen. Clients preisen dazu außer den tatsächlich unterstützten auch zusätzlich zufällig ausgewürfelte, nicht existente Versionen an. Die Erfinder des Verfahrens nennen das dann GREASE (Generate Random Extensions And Sustain Extensibility) – was auf Deutsch Schmiere bedeutet. Und nein, der GREASE-Draft wurde nicht im April veröffentlicht.